IEEE Xplore® RELEASE 2.1

Welcome United States Patent and Trademark Office

Wed, 14 Dec 2005, 2:55:03 PM EST

Edit an existing query or compose a new query in the Search Query Display.

Select a search number (#) to:

- Add a query to the Search Query Display
- Combine search queries using AND, OR, or NOT
- Delete a search
- Run a search

Search Query Display

[redacted]

Recent Search Queries

#1 ((interrupt service routine)<in>metadata)

#2 (error handling routine<IN>metadata)

#3 (safe state<IN>metadata)

#4 (safe state<IN>metadata)

#5 (define state<IN>metadata)

#6 (stable state<IN>metadata)

#7 (stable state<in>metadata) and (unused memory)

#8 (unused memory<IN>metadata)

#9 (irregular state<IN>metadata)

[redacted]

Indexed by Inspec®

Terms used
program memory defined or **stable** or **safe state interrupt error routine**

Found **64** of **167,655**

Sort results by relevance

Display results expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 64 Result page: **1** 2 3 4 next

Relevance scale

1 Special issue: Game-playing programs: theory and practice

M. A. Bramer
April 1982 **ACM SIGART Bulletin**, Issue 80
Publisher: ACM Press
Full text available: pdf(9.23 MB) Additional Information: full citation, abstract

This collection of articles has been brought together to provide SIGART members with an overview of Artificial Intelligence approaches to constructing game-playing programs. Papers on both theory and practice are included.

2 The family of concurrent logic programming languages

Ehud Shapiro
September 1989 **ACM Computing Surveys (CSUR)**, Volume 21 Issue 3
Publisher: ACM Press
Full text available: pdf(9.62 MB) Additional Information: full citation, abstract, references, citings, index terms

Concurrent logic languages are high-level programming languages for parallel and distributed systems that offer a wide range of both known and novel concurrent programming techniques. Being logic programming languages, they preserve many advantages of the abstract logic programming model, including the logical reading of programs and computations, the convenience of representing data structures with logical terms and manipulating them using unification, and the amenability to metaprogrammin ...

3 A structural view of the Cedar programming environment

Daniel C. Swinehart, Polle T. Zellweger, Richard J. Beach, Robert B. Hagmann
August 1986 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 8 Issue 4
Publisher: ACM Press
Full text available: pdf(6.32 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents an overview of the Cedar programming environment, focusing on its overall structure—that is, the major components of Cedar and the way they are organized. Cedar supports the development of programs written in a single programming language, also called Cedar. Its primary purpose is to increase the productivity of programmers whose activities include experimental programming and the development of prototype

software systems for a high-performance personal computer. T ...

4 Distributed operating systems

Andrew S. Tanenbaum, Robbert Van Renesse
December 1985 **ACM Computing Surveys (CSUR)**, Volume 17 Issue 4
Publisher: ACM Press

Full text available: pdf(5.49 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Distributed operating systems have many aspects in common with centralized ones, but they also differ in certain ways. This paper is intended as an introduction to distributed operating systems, and especially to current university research about them. After a discussion of what constitutes a distributed operating system and how it is distinguished from a computer network, various key design issues are discussed. Then several examples of current research projects are examined in some detail ...

5 Software safety: why, what, and how

Nancy G. Leveson
June 1986 **ACM Computing Surveys (CSUR)**, Volume 18 Issue 2
Publisher: ACM Press

Full text available: pdf(4.18 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Software safety issues become important when computers are used to control real-time, safety-critical processes. This survey attempts to explain why there is a problem, what the problem is, and what is known about how to solve it. Since this is a relatively new software research area, emphasis is placed on delineating the outstanding issues and research topics.

6 Third Generation Computer Systems

Peter J. Denning
December 1971 **ACM Computing Surveys (CSUR)**, Volume 3 Issue 4
Publisher: ACM Press

Full text available: pdf(3.52 MB) Additional Information: full citation, abstract, references, citings, index terms

The common features of third generation operating systems are surveyed from a general view, with emphasis on the common abstractions that constitute at least the basis for a "theory" of operating systems. Properties of specific systems are not discussed except where examples are useful. The technical aspects of issues and concepts are stressed, the nontechnical aspects mentioned only briefly. A perfunctory knowledge of third generation systems is presumed.

7 Distributed systems - programming and management: On remote procedure call

Patrícia Gomes Soares
November 1992 **Proceedings of the 1992 conference of the Centre for Advanced Studies on Collaborative research - Volume 2**
Publisher: IBM Press

Full text available: pdf(4.52 MB) Additional Information: full citation, abstract, references, citings

The Remote Procedure Call (RPC) paradigm is reviewed. The concept is described, along with the backbone structure of the mechanisms that support it. An overview of works in supporting these mechanisms is discussed. Extensions to the paradigm that have been proposed to enlarge its suitability, are studied. The main contributions of this paper are a standard view and classification of RPC mechanisms according to different perspectives, and a snapshot of the paradigm in use today and of goals for t ...

8 Mondrix: memory isolation for linux using mondriaan memory protection

Emmett Witchel, Junghwan Rhee, Krste Asanović
October 2005 **ACM SIGOPS Operating Systems Review , Proceedings of the twentieth ACM symposium on Operating systems principles SOSP '05**, Volume 39 Issue 5
Publisher: ACM Press
Full text available: pdf(332.09 KB) Additional Information: full citation, abstract, references, index terms

This paper presents the design and an evaluation of Mondrix, a version of the Linux kernel with Mondriaan Memory Protection (MMP). MMP is a combination of hardware and software that provides efficient fine-grained memory protection between multiple protection domains sharing a linear address space. Mondrix uses MMP to enforce isolation between kernel modules which helps detect bugs, limits their damage, and improves kernel robustness and maintainability. During development, MMP exposed two kerne ...

Keywords: fine-grained memory protection

9 A coherent distributed file cache with directory write-behind

Timothy Mann, Andrew Birrell, Andy Hisgen, Charles Jerian, Garret Swart
May 1994 **ACM Transactions on Computer Systems (TOCS)**, Volume 12 Issue 2
Publisher: ACM Press
Full text available: pdf(3.21 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Extensive caching is a key feature of the Echo distributed file system. Echo client machines maintain coherent caches of file and directory data and properties, with write-behind (delayed write-back) of all cached information. Echo specifies ordering constraints on this write-behind, enabling applications to store and maintain consistent data structures in the file system even when crashes or network faults prevent some writes from being completed. In this paper we describe ...

Keywords: coherence, file caching, write-behind

10 Digital control of industrial processes

Cecil L. Smith
September 1970 **ACM Computing Surveys (CSUR)**, Volume 2 Issue 3
Publisher: ACM Press
Full text available: pdf(2.11 MB) Additional Information: full citation, references, citings, index terms

11 An interactive graphical display monitor in a batch-processing environment with remote entry

Alan H. Bond, Jerry Rightnour, L. Steven Coles
November 1969 **Communications of the ACM**, Volume 12 Issue 11
Publisher: ACM Press
Full text available: pdf(1.38 MB) Additional Information: full citation, abstract, references, citings, index terms

A graphic monitor program is described. It was developed at Carnegie-Mellon University for the CDC G21 computer, which is a general purpose, batch-processing system with remote entry. The existing G21 system and the graphics hardware are described. The graphic monitor is a resident auxiliary monitor which provides comprehensive managerial capability over the graphical system in response to commands from the human user. It also will respond to commands from a user program through a similar i ...

Keywords: design of graphical system, graphic interface, graphic monitor, graphics, graphics in batch environment, man/machine interaction

12 The space shuttle primary computer system

Alfred Spector, David Gifford
September 1984 **Communications of the ACM**, Volume 27 Issue 9
Publisher: ACM Press
Full text available: pdf(5.34 MB) Additional Information: full citation, references, citings, index terms

Keywords: PASS, avionics system, space shuttle

13 Implementation of Argus

B. Liskov, D. Curtis, P. Johnson, R. Scheifer
November 1987 **ACM SIGOPS Operating Systems Review , Proceedings of the eleventh ACM Symposium on Operating systems principles SOSP '87**, Volume 21 Issue 5
Publisher: ACM Press
Full text available: pdf(1.34 MB) Additional Information: full citation, abstract, references, citings, index terms

Argus is a programming language and system developed to support the construction and execution of distributed programs. This paper describes the implementation of Argus, with particular emphasis on the way we implement atomic actions, because this is where Argus differs most from other implemented systems. The paper also discusses the performance of Argus. The cost of actions is quite reasonable, indicating that action systems like Argus are practical.

14 Introducing Ada 9X

John Barnes
November 1993 **ACM SIGAda Ada Letters**, Volume XIII Issue 6
Publisher: ACM Press
Full text available: pdf(4.39 MB) Additional Information: full citation, citings, index terms

15 CONS should not CONS its arguments, or, a lazy alloc is a smart alloc

Henry G. Baker
March 1992 **ACM SIGPLAN Notices**, Volume 27 Issue 3
Publisher: ACM Press
Full text available: pdf(1.52 MB) Additional Information: full citation, abstract, index terms

Lazy allocation is a model for allocating objects on the execution stack of a high-level language which does not create dangling references. Our model provides safe transportation into the heap for objects that may survive the deallocation of the surrounding stack frame. Space for objects that do not survive the deallocation of the surrounding stack frame is reclaimed without additional effort when the stack is popped. Lazy allocation thus performs a first-level garbage collection, and if ...

16 Session summaries from the 17th symposium on operating systems principle (SOSP'99)

Jay Lepreau, Eric Eide

April 2000 **ACM SIGOPS Operating Systems Review**, Volume 34 Issue 2
Publisher: ACM Press
Full text available: pdf(3.15 MB) Additional Information: full citation, index terms

17 Continuous program optimization: A case study

Thomas Kistler, Michael Franz
July 2003 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 25 Issue 4
Publisher: ACM Press
Full text available: pdf(877.67 KB) Additional Information: full citation, abstract, references, index terms, review

Much of the software in everyday operation is not making optimal use of the hardware on which it actually runs. Among the reasons for this discrepancy are hardware/software mismatches, modularization overheads introduced by software engineering considerations, and the inability of systems to adapt to users' behaviors.A solution to these problems is to delay code generation until load time. This is the earliest point at which a piece of software can be fine-tuned to the actual capabilities of the ...

Keywords: Dynamic code generation, continuous program optimization, dynamic reoptimization

18 The Flux OSKit: a substrate for kernel and language research

Bryan Ford, Godmar Back, Greg Benson, Jay Lepreau, Albert Lin, Olin Shivers
October 1997 **ACM SIGOPS Operating Systems Review , Proceedings of the sixteenth ACM symposium on Operating systems principles SOSP '97**, Volume 31 Issue 5
Publisher: ACM Press
Full text available: pdf(2.47 MB) Additional Information: full citation, references, citings, index terms

19 Draft report on requirements for a common prototyping system

R. P. Gabriel
March 1989 **ACM SIGPLAN Notices**, Volume 24 Issue 3
Publisher: ACM Press
Full text available: pdf(4.76 MB) Additional Information: full citation, citings, index terms

20 Higher-order distributed objects

Henry Cejtin, Suresh Jagannathan, Richard Kelsey
September 1995 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 17 Issue 5
Publisher: ACM Press
Full text available: pdf(2.33 MB) Additional Information: full citation, abstract, references, citings, index terms, review

We describe a distributed implementation of Scheme that permits efficient transmission of higher-order objects such as closures and continuations. The integration of distributed communication facilities within a higher-order programming language engenders a number of new abstractions and paradigms for distributed computing. Among these are user-specified load-balancing and migration policies for threads, incrementally linked distributed computations, and parameterized client-server applicat ...

Keywords: concurrency, continuations, higher-order languages, message-passing

Results 1 - 20 of 64 Result page: **1** 2 3 4 next

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	5	(Uwe near Daemmrich).in.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/12/14 11:32
L2	18	(Dieter near Buchholz).in.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/12/14 11:33
L3	2	1 and 2	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/12/14 11:33
L4	50731	"714"/$.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/12/14 11:33
L5	27274	"711"/$.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/12/14 11:34
L6	27274	"711"/.clas.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/12/14 11:35
L7	42332	(define$2 or stable or safe) near state	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/12/14 11:35
L8	66009	(define$2 or stable or safe) near2 state	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/12/14 11:36
L9	1989	((define$2 or stable or safe) near2 state) same (restor$4 or recover$5)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/12/14 11:36
L10	4629	interrupt adj service adj routine	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/12/14 11:37

L11	554	error adj handling adj routine	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/12/14 11:37
L12	884	(irregular or undefine$4) adj state	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/12/14 11:38
L13	9163	unuse$4 near2 (area or location or space or segment$2)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/12/14 11:40
L14	5160	10 or 11	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/12/14 11:40
L15	59	9 and 14	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/12/14 12:15
L16	2	13 and 15	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/12/14 11:41
L17	1	12 and 15	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/12/14 11:41
L18	75935	4 or 5	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/12/14 12:15
L19	2337	8 and 18	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/12/14 12:18
L20	19	12 and 19	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/12/14 12:18